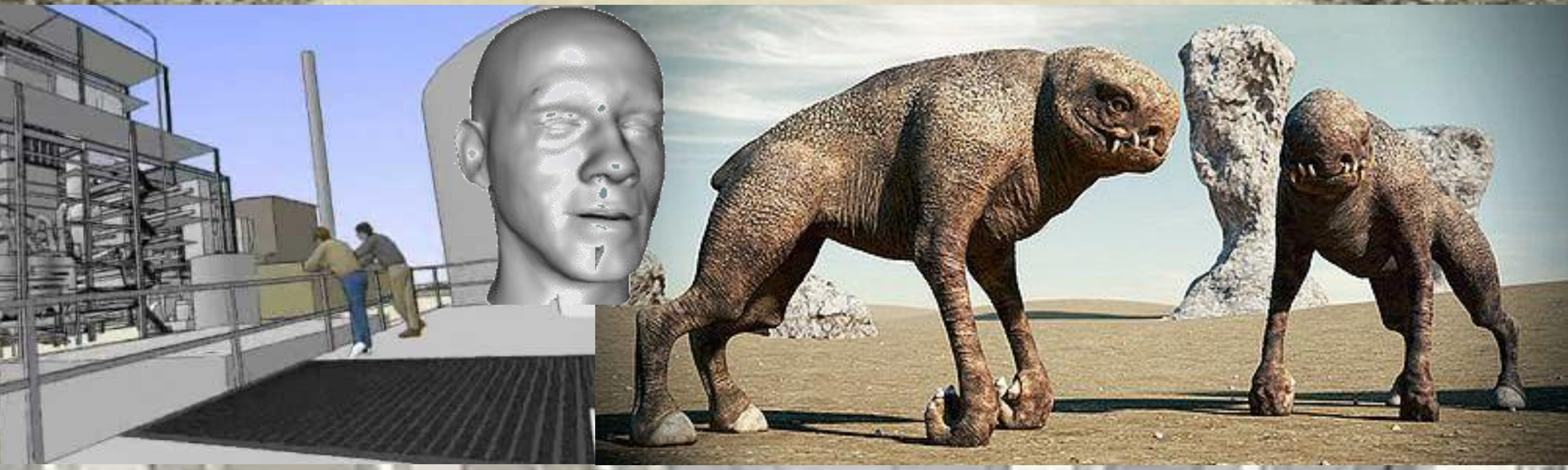
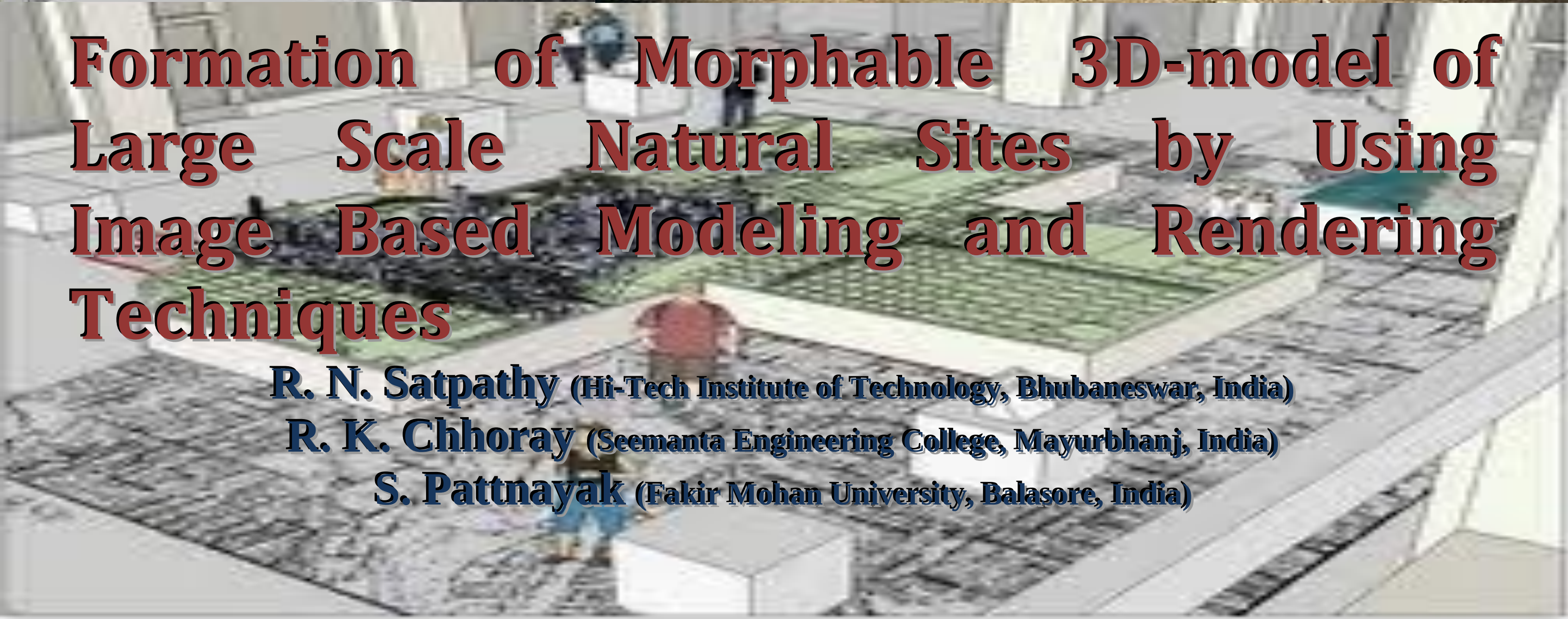




Formation of Morphable 3D-model of Large Scale Natural Sites by Using Image Based Modeling and Rendering Techniques

R. N. Satpathy (Hi-Tech Institute of Technology, Bhubaneswar, India)
R. K. Chhoray (Seemanta Engineering College, Mayurbhanj, India)
S. Pattnayak (Fakir Mohan University, Balasore, India)

International Research Journal
ISSN 1839-6518

Vol. 1, No. 2, 2011
82800102201103

Formation of Morphable 3D-model of Large Scale Natural Sites by Using Image Based Modeling and Rendering Techniques

R. N. Satpathy (Hi-Tech Institute of Technology, Bhubaneswar, India)
R. K. Chhoray (Seemanta Engineering College, Mayurbhanj, India)
S. Pattnayak (Fakir Mohan University, Balasore, India)

IARS' International Research Journal
ISSN 1839-6518

Vol. 1, No. 2, 2011
82800102201103

Formation of Morphable 3D-model of Large Scale Natural Sites by Using Image Based Modeling and Rendering Techniques

Abstract

No global 3D model of the environment needs to be assembled, a process which can be extremely cumbersome and error prone for large scale scenes e.g. the global registration of multiple local models can accumulate a great amount of error, while it also presumes a very accurate extraction of the underlying geometry. On the contrary, neither any such accurate geometric reconstruction of the individual local 3D models nor a very precise registration between them is required by our framework in order that it can produce satisfactory results. This paper presents an application of LP based MRF optimization techniques and also we have turned our attention to a different re- search topic: the proposal of novel image based modeling and rendering methods, which are capable of automatically reproducing faithful (i.e. photorealistic) digital copies of complex 3D virtual environments, while also allowing the virtual exploration of these environments at interactive frame rates.

Keywords- MRF , LP,SEARCH, MORPHING

Introduction

In recent years, there has been a great deal of new developments in applying topological tools to image analysis. In particular, computing topological invariants has been of great importance in understanding the shape of an arbitrary 2-dimensionall (2D) or 3-dimensional (3D) object. The most powerful invariant of these objects is the fundamental group. Unfortunately, fundamental groups are highly non-commutative and difficult to work w. In fact, the general problem in determining whether two given groups are isomorphic is not decidable (meaning that there is no algorithm can solve the problem). For fundamental groups of 3D objects, this problem is decidable but no practical algorithm has been found yet. As a result, homology groups have received the most attention because their computations are more feasible and they still provide significant information about the shape of the object. This leads to the motivating problem addressed in this paper: Given a 3D object in 3- dimensional Euclidean space R3, determine homology groups of the object in the most effective way by only analyzing the digitization of the object. The properties of homology groups have applications in many areas of bioinformatics and image processing. We particularly look at a set of points in 3D digital space, and our purpose is to find homology groups of the data set.

One research problem of computer graphics that has attracted a lot of attention over the last years is the creation of modeling and rendering systems capable to provide photorealistic & interactive walkthroughs of complex, real world environments. Two are the main approaches that have been proposed so far for that purpose. On one hand, there exist those techniques that are geometry based techniques i.e. first try to estimate an accurate global 3D model of the scene. Then they use the extracted 3D model in order to render the scene under any given viewpoint. One of their advantages is that they provide great flexibility and allow many of the scene's properties to be modified during rendering. E.g. by having a global 3D model one can readily alter not only the viewpoint, but also the lighting conditions of the scene. However, their big disadvantage comes from the fact that extracting an accurate global 3D model can be either extremely time consuming or very difficult (not to say impossible) in many cases. For example such a 3D-model construction task can be easy for scenes containing mostly planar objects (e.g. architectural type scenes), but becomes extremely hard for outdoor scenes containing objects with irregular geometry e.g. trees. The automatic extraction of a 3D-model from images, also known as multiple view geometry, has been (and still is) an active research topic in computer vision. In fact a significant amount of progress has been achieved in this area over the last years.

Overview of the modeling pipeline

A diagram of our system's modeling pipeline is shown in Figure 1. We will first consider the simpler case of having only one stereoscopic view per key- position of the path. Prior to capturing these stereoscopic views, a calibration of the stereoscopic camera needs to take place first. During this stage both the external parameters (i.e. the relative 3D rotation and translation between the left and right camera), as well as the internal parameters of the stereoscopic camera are estimated. We make the common assumption that both the left and right cameras are modeled by the usual pinhole. In this case their internal parameters are contained in the so called intrinsic matrices K_{left}, K_{right}. Any such matrix has the following form:

$$\begin{bmatrix} f_x & c & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here (f_x, f_y) represents the focal length, c describes the skewness of the 2 image axes while (u_0, v_0) represents the principal point. We also model (both radial and tangential) lens distortion and the following model is assumed for this purpose:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = \begin{bmatrix} 1 + d_1 r^2 + d_2 r^4 + d_5 r^6 + 2d_3 xy + d_4(r^2 + 2x^2) \\ 1 + d_1 r^2 + d_2 r^4 + d_5 r^6 + d_3(r^2 + 2y^2) + 2d_4 xy \end{bmatrix}$$

Here (x, y) are the ideal (distortion free) pixel coordinates, $(\hat{x}, \hat{y})$ are the corresponding observed image coordinates and $r = \sqrt{x^2 + y^2}$. For estimating all of these parameters we apply a method similar to that in [3], using as input stereoscopic image pairs of a calibrated chess pattern captured at random positions and orientations by our camera (see Figure 2). After the camera calibration has finished, the following stages of the modeling pipeline need to take place:

1. Local 3D models construction: A photometric and geometric representation of the scene near each key position of the path is constructed. The geometric part of a local model needs to be only an approximation of the true scene geometry.
2. Approximate registration between successive local 3D models: An estimation of the relative pose between successive local models takes place here. We should note that only a coarse estimate of the relative pose is needed, since this will not be used for an exact registration of the local models, but merely for the morphing procedure that takes place later.
3. 3D morph able models construction: The photometric as well as the geometric morphing between successive local 3D models is estimated during this stage of the modeling pipeline.

In the case that there are multiple views per key position of the path, then, as already explained, there will also have to be an additional stage responsible for the 3D-mosaics construction. This stage needs to take place prior to the registration step and is described. Finally, we describe the rendering pipeline of our system.

Local 3D models construction

For each stereoscopic image pair, a 3D model describing the scene locally (i.e. as seen from the camera viewpoint) must be produced during this stage. To this end, a stereo matching procedure is applied to the left and right images (denoted I_{left} and I_{right}), so that disparity can be estimated for all points inside a selected image region dom_0 of I_{left}.Using then the resulting disparity map (as well as the calibration matrices of the cameras) a 3D reconstruction takes place and thus the maps $X_0, Y_0 and Z_0$ are produced (see Figure 3). These maps respectively contain the x, y and z coordinates of the reconstructed points with respect to the 3D coordinate system of the left camera. The set $L_0 = (X_0, Y_0, Z_0, I_{left}, dom_0)$ consisting of the images X_0, Y_0, Z_0 (the geometric-maps), the image region dom0 (valid domain of geometric maps) and the image I_{left} (the photometric map) makes up what we call a ``local model'' L_0 . Hereafter that term will implicitly refer to such a set of elements. By applying a 2D triangulation on the image grid of a local model, a textured 3D triangle mesh can be produced. The 3D coordinates of triangle vertices are obtained from the underlying geometric maps while texture is obtained from I_{left} and mapped onto the mesh (see Figure 4). It should be noted that the geometric maps of a local model are expected to contain only an approximation of the scene's true geometric model.

Disparity estimation

Disparity estimation proceeds in two stages (see Figure 5). During the first stage, we reduce the problem of stereo matching to a discrete labeling problem which is going to be solved through the energy optimization a 1st order Markov Random Field. The nodes of the corresponding MRF are going to be the pixels of the left image and the single node potential for assigning disparity d_p to pixel p is going to be estimated as follows:

$$V_p(d_p) = \left| I_{right}(p - d_p) - I_{left}(p) \right|^2$$

Furthermore, for the pair wise potentials the truncated semi-metric distance between disparities has been used, i.e.:

$$V_{pq}(d_p, d_q) = \min\left(\lambda_{0,} \left| d_p - d_q \right|^2 \right)$$

Here λ_0 denotes the maximum allowed penalty that can be imposed. For the optimization of the above MRF, the LP based algorithms have been used, since they can always guarantee a solution which is close to the optimal one. The role of the first stage is to produce a good initial estimate of the disparity and to avoid any bad local minima during the optimization process. Its output is then given as input to the next stage of the disparity estimation process, where a global refinement of the disparity field is taking place. To this end, the energy of a first order Markov Random Field is again being minimized .The difference, however, with respect to the first stage, is that now a local continuous optimization scheme is being used so that disparities with sub-pixel accuracy can be obtained. In particular, we use a standard gradient descent type algorithm for minimizing the following energy function:

$\sum_{(i,j)} I_{right}(i - d_{ij}, j) - I_{left}(i, j)^2 + \lambda \sum_{(i,j)} \sum_{p \in N_{ij}} g(d_{ij} - d_p)$ where N_{ij} is the 4-point neighborhood of pixel (i, j) and d_{ij} again represents the unknown disparity field. The λ parameter is a regularization parameter, while the potential function g(.) is chosen to be discontinuity adaptive [2] (e.g. a truncated quadratic distance), so that a regularized solution, which also preserves discontinuities, is finally computed. The disparity field is initialized with the values estimated during the first stage. Due to this initialization the gradient descent algorithm usually converges very fast and does not get trapped to any poor local minima.

Relative pose estimation between successive local models

Let $L_k = (X_k, Y_k, Z_k, I_k, dom_k)$ and $L_{k+1} = (X_{k+1}, Y_{k+1}, Z_{k+1}, I_{k+1}, dom_{k+1})$ be 2 successive local models along the path. For their relative pose estimation, we need to extract a set of point matches (p_i, q_i) between the left images I_k, I_{k+1} of models L_k, L_{k+1} respectively. Assuming that such a set of matches already exists, the pose estimation can proceed as follows: the 3D points of L_k corresponding to p_i are $P_i = (X_k(p_i), Y_k(p_i), Z_k(p_i))$ and so there projections of p_i on image I_{k+1} are:

$p_i^{'} = K_{left}(R.P_i + T) \in P^2$, where R (a 3 X3 ortho-normal matrix) and T (a 3D vector) represent the unknown rotation and translation respectively.

So the pose estimation can be achieved by minimizing the following re-projection error:

$$\sum_i dist(q_i, p_i^{'})^2$$

where dist denotes Euclidean image distance. For this purpose, an iterative constrained minimization algorithm may be applied with rotation represented internally by a quaternion $q(\|q\| = 1)$. The essential matrix (also computable by the help of the matches (p_i, q_i) and $K_{left}, K_{right})$ can be used to provide an initial estimate [6] for the iterative algorithm.

Wide-baseline feature matching under camera looming

Therefore the pose estimation problem is reduced to that of extracting a sparse set of correspondences between I_K, I_{K+1} . A usual method for tackling the latter problem is the following: first a set of interest points in I_K are extracted (using an interest point detector). Then for each interest point, say p, a set of candidate points CANDp inside a large rectangular region SEARCHp of I_{K+1} are examined and the best one is selected according to a similarity measure. Usually the candidate points are extracted by applying an interest point detector to region SEARCHp as well.

However unlike left/right images of a stereoscopic view, I_K, I_{K+1} are separated by a wide baseline. Simple measures like correlation have been proved extremely inefficient in such cases. Assuming a smooth predefined path (and therefore a smooth change in orientation between I_K, I_{K+1}), it is safe to assume that the main difference at an object's appearance in images I_K, I_{K+1}, comes from the forward camera motion along the Z axis (looming). The idea for extracting valid correspondences is then based on the following observation: the dominant effect of an object being closer to the camera in image I_{K+1} is that its image region in I_{K+1} appears scaled by a certain scale factor s>1. That is, if $p \in I_K, q \in I_{K+1}$ are corresponding pixels: $I_{k+1}(sq) \approx I_k(p)$. So an image patch of I_K at p should look similar to an image patch of an appropriately rescaled (by S^{-1}) version of I_{K+1}. Of course, the scale factor s varies across the image. Therefore the following strategy, for extracting reliable matches, can be applied:

1. Quantize the scale space of s to a discrete set of values $S = \{s_j\}_{j=0}^{n}$, where $1 = s_0 < s_1 < .... < s_n$
2. Rescale I_{K+1} by the inverse scale s_j^{-1} for all $s_j \in S$ to get rescaled images I_{K+1,s_j}

For any $q \in I_{k+1,} p \in I_k$, let us denote by $I_{K+1,s_j}(q)$ a (small) fixed-size patch around the projection of q on I_{K+1,s_j} and by $\overline{I_k(p)}$ an equal-size patch of I_k at p.

3. Given any point $p \in I_K$ and its set of candidate points $CAND_P = \{q_i\}$ in $I_{k+1,}$ use correlation to find among the patches at any q_i and across any scale s_i, the one most similar to I_k at p :

$$(q_i, s_i) = \arg \max_{q_i, s_j} corr\left(\overline{I_{k+1,s_j}(q_i)}, \overline{I_k(p)}\right)$$

This way, apart from a matching point $q^{'} \in I_{k+1}$, a scale estimate $s^{'}$ is provided for point p as well.

The above strategy has been proved very effective, giving a high percentage of exact matches even in cases with very large looming. Such an example can be seen in Figure 6 wherein the images baseline is ≈ 15 meters, resulting in scale factors of $size \approx 2.5$ for certain image regions. Even if we set as candidate points CANDp of a point p, all points inside SEARCHp in the in the other image (and not only detected Interest points therein), the above procedure still picks the right matches in most cases. The results in Figure 7 have been produced in this way.

Morphing estimation between successive local models

At the current stage of the modeling pipeline, a series of approximate local 3D models (along with approximate estimates of the relative pose between every successive two) are available to us. Let $L_k = (X_k, Y_k, Z_k, I_k, dom_k)$, $L_{k+1} = (X_{k+1}, Y_{k+1}, Z_{k+1}, I_{k+1}, dom_{k+1})$ be such a pair of successive local models and pos_k to pos_{k+1} their corresponding key positions on the path. By making use of the approximate pose estimate between L_k and L_{k+1}, we will assume hereafter that the 3D vertices of both models are expressed in a common 3D coordinate system. Rather than trying to create a consistent global model by combining all local ones (a rather tedious task requiring among others high quality geometry and pose estimation) we will instead follow a different approach, which is based on the following observation: near path point pos_k, model L_k is ideal for representing the surrounding scene. On the other hand, as we move forward along the path approaching key position of the next model L_{k+1}, the photometric and geometric properties of the environment are much better captured by that model. (For example compare the fine details of the rocks that are revealed in Figure 6 and are not visible in Figure 7). So during transition from pos_k to pos_{k+1} , we will try to gradually morph model L_k into a new destination model, which should coincide with L_{k+1} upon reaching point posk+1. (In fact, only part of this destination model can coincide with L_{k+1} since in general L_k, L_{k+1} will not represent exactly the same part of the scene). This morphing should be geometric as well as photometric (the latter wherever possible) and should proceed in a physically valid way. For this reason, we will use what we call a "morph able 3D-model ".

$$L_{morh} = L_k \cup (X_{dst}, Y_{dst}, Z_{dst}, I_{dst})$$

In addition to including the elements of L_k, L_{morh} also consists of map $X_{dst}, Y_{dst}, Z_{dst}$ and map I_{dst} containing respectively the destination 3D vertices and destination color values for all points of L_k. At any time during the rendering process, the 3D coordinates $vert_{ij}$ and color col_{ij} of the vertex of L_{morh} at point (i, j) will then be:

$$vert_{ij} = \begin{bmatrix} (1-m)X_k(i,j) + mX_{dst}(i,j) \\ (1-m)Y_k(i,j) + mY_{dst}(i,j) \\ (1-m)Z_k(i,j) + mZ_{dst}(i,j) \end{bmatrix} \ldots\ldots(1)$$

$$col_{ij} = (1-m)I_k(i,j) + mI_{dst}(i,j) \qquad \ldots\ldots\ldots(2)$$

where m is a parameter determining the amount of morphing (m =0 at pos_k, m=1 at pos_{k+1} and 0<m<1 in between) . Specifying therefore L_{morh} amounts to filling in the values of the destination maps $\{X, Y, Z, I\}_{dst}$ for each point $p \in dom_k$. For this purpose, a 2-step procedure will be followed that depends on whether point p has a physically corresponding point in L_{k+1} or not:

1. Let be that subset of region $dom_k \subseteq I_k$, consisting only of those L_k points that have physically corresponding points in model L_{k+1} and let $u_{k \to k+1}$ be a function which maps these points to their counterparts in the I_{k+1} image.

(Region ψ represents that part of the scene which is common to both models $L_{k,} I_{k+1})$. Since model L_k (after morphing) should coincide with L_{k+1}, it must then hold:

$$\begin{bmatrix} X_{dst}(p) \\ Y_{dst}(p) \\ Z_{dst}(p) \\ I_{dst}(p) \end{bmatrix} = \begin{bmatrix} X_{k+1}(u_{k \to k+1}(p)) \\ Y_{k+1}(u_{k \to k+1}(p)) \\ Z_{k+1}(u_{k \to k+1}(p)) \\ I_{k+1}(u_{k \to k+1}(p)) \end{bmatrix} \forall p \in \psi \text{ .....(3)}$$

Points of region ψ are therefore transformed both photo metrically and geometrically.

2. The rest of the points (that is points in $\psi = dom_k \setminus \psi)$ do not have counterparts in model L_{k+1}. So these points will retain their color value (from model L_k) at the destination maps and no photometric morphing will take place:

$$I_{dst}(p) = I_k(p), \forall p \in \overline{\psi} \quad \text{....................(4)}$$

But we still need to apply geometric morphing to those points so that no Distortion / discontinuity in the 3D structure are observed during transition from pos_k to pos_{k+1}. Therefore we still need to fill in the destination 3D coordinates for all points in $\overline{\psi}$. The 2 important remaining issues (which also constitute the core of the morphing procedure) are:

* How to compute the mapping $u_{k \to k+1}$. This is equivalent to estimating a 2D optical flow field between the left images I_k and I_{k+1}.

* And how to obtain the values of the destination geometric maps at the points

inside region $\overline{\psi}$, needed for the geometric morphing therein. Both of these issues will be the subject of the two subsections that follow.

Estimating optical flow between wide baseline images I_k and I_{k+1}

In general, obtaining a reliable, relatively dense optical flow field between wide baseline images like I_k and I_{k+1} is a particularly difficult problem. Without additional input, usually only a sparse set of optical flow vectors can be obtained in the best case. In this case the basic problems are:

1. For every point in I_k, a large region of image I_{k+1} has to be searched for obtaining a corresponding point. This way the chance of an erroneous optical flow vector increases significantly (as well as the computational cost) .
2. Simple measures (like correlation) are very inefficient for comparing pixel blocks between wide baseline images .
3. Even if both of the above problems are solved, optical flow estimation is inherently an ill posed problem and additional assumptions are needed. In particular, we need to somehow impose the condition that the optical flow field will be piecewise smooth.

For dealing with the first problem, we will make use of the underlying geometric maps X_k, Y_k, Z_k of model L_k as well as the relative pose between I_k and I_{k+1}. By using these quantities, we can theoretically re-project any point, say p, of I_k onto image I_{k+1}. In practice since all of the above quantities are estimated only approximately, this permits us just to restrict the searching over a smaller region R_p around the re-projection point. The search region can be restricted further by taking the intersection of R_p with a small zone around the epipolar line corresponding to p. In addition, since we are interested in searching only for points of I_{k+1} that belong to dom_{k+1} (this is where L_{k+1} is defined), the final search region SEARCHp of p will be $R_p \cap dom_{k+1}$. If SEARCHp is empty, then no optical flow vector will be estimated and point p will be considered as not belonging to region ψ . For dealing with the second problem, we will use a technique similar to get a sparse set of correspondences. As already stated therein, the dominant effect due to a looming of the camera is that pixel neighborhoods in image I_{k+1} are scaled by a factor varying across the image. The solution proposed therein was to compare image patches of I_k not only with patches from I_{k+1}, but also with patches from rescaled versions of the latter image. We will use the same technique here, with the only difference being that instead of doing that for a sparse group of features we will now apply it to a dense set of pixels of image I_k. For this purpose we will again use a discrete set of scale factors $S = \{1 = s_0 < s_1 < ..... < s_n\}$ and we will rescale image I_{k+1} by each one of these factors where, as before, image I_{k+1} rescaled by s^{-1} (with $s \in S)$ will be denoted by $I_{K+1,s}$. As we shall see in the next paragraph, this will have the effect of having to change the type of labels that we will use in the associated labeling problem.

Finally, to deal with the ill posed character of the problem, we will first reduce the optical flow estimation to a discrete labeling problem and then formulate it in terms of minimizing the energy of a first order Markov Random Field [9]. What is worth noting here is that, contrary to a standard optical flow estimation procedure, the labels will now consist of vectors , $\ell = (d_x, d_y, s) \in R^2 \times S$ where the first 2 coordinates denote the components of the optical flow vector while the third one denotes the scale factor. This means that after labeling, not only an optical flow, but also scale estimation will be provided for each point (see Figure 9). Given a label l, we will denote its optical flow vector by $flow(\ell) = (d_x, d_y)$ and its scale by $scale(\ell) = s$. Based on what was already mentioned above, the labels which are allowed to be assigned to a point p in I_k will be coming from the following set: $LABELS_P = \{q - p : q \in SEARCH_P\} \times S$. This definition of the label set $SEARCH_P$ simply encodes the following two things:

* For any point p of the first image, we are searching for corresponding points q only inside the restricted region $SEARCH_P$

* We also search across all scales in S, i.e. given a candidate matching point $q \in SEARCH_P$ for p, we compare patch $\overline{I_k(p) \in I_k)}$ with any of the patches $\overline{I_{k+1,s}(q)} \in I_{k+1,s}$;s where the scale s traverses all the elements of set S (see Figure 8). As before $\overline{I_k(p)} \in I_k)$ denotes a fixed size patch around p, while $\overline{I_{k+1,s}(q)} \in I_{k+1,s}$ denotes an equal size patch, which is located around the projection of q on the rescaled image $I_{K+1,s}$.

Getting an optical flow field is then equivalent to picking one element from the Cartesian product $LABELS = \prod_{p\in\psi} LABELS_P$. In our case, that element x of LABELS, which minimizes the following energy should be chosen:

$$f(x) = \sum_{(p,p')\in N} V_{pp'}(x_p, x_{p'}) + \sum_{p\in\psi} V_p(x_p) \;.......(5)$$

The first sum in F(x) represents the prior term and penalizes optical flow fields which are not piecewise smooth, while the second sum in the above energy represents the likelihood and measures how well the corresponding optical flow agrees with the observed image data. The symbol N denotes a set of interacting pairs of pixels inside ψ . (we typically assume a 4-system neighborhood) and $V_{pp'}(.,.)$ denotes the pair wise potential function of the MRF. In our case, this function can be set as follows:

$$V_{pp'}(x_p, x_{p'}) = \min\left(\left\| flow(x_p) - flow(x_{p'})\right\|^2 + \left|scale(x_p) - scale(x_{p'})\right|^2, \lambda_0\right) \quad ..(6)$$

where λ_0 denotes the maximum pair wise penalty that can be imposed. Simpler pair wise potential functions, like the Potts function, have been also tested. Regarding the terms $V_p(x_p)$, these measure the correlation between corresponding image patches as determined by the labeling x. According to a labeling x, for a point p in I_k its corresponding point is the projection into image $I_{k+1,scale(x_p)}$ of point $p + flow(x_p)$. This means that we should compare the patches $\overline{I_k(p)}$ and $\overline{I_{k+1,scale}(x_p)(p + flow(x_p))}$ and, for this reason, we set:

$$V_p(x_p) = corr\left(\overline{I_k(p)}, \overline{I_{k+1,scale}(x_p)(p + flow(x_p))}\right)(7)$$

The above energy F(x) can be minimized using any of the LP-based MRF optimization algorithms . The resulting optical flow, obtained when using the two images of Figure 6 as input, is shown in Figure 9. For comparison, we also show there (Figure 9) the corresponding optical flow result, which is estimated if no search across scales takes place i.e. $S = \{1\}$. As expected, in this case, the resulting optical flow is very noisy for regions that are actually undergoing a large change of scale.

Geometric morphing in region $\overline{\psi}$

After estimation of optical flow $u_{k \to k+1}$, we may apply equation (3) to all points in ψ and thus fill the arrays $X_{dst}, Y_{dst}, Z_{dst}$ therein (see Figure 10). Therefore, at this stage of the modeling pipeline, the values of the destination geometric maps $X_{dst}, Y_{dst}, Z_{dst}$ are known for all points inside region ψ , but are unknown for all points inside region $\overline{\psi} = dom_k \setminus \psi$ (i.e. the region which is the complement of ψ in dom_k). Hereafter, the already known values of the destination geometric maps will be denoted by $\hat{X}_{dst}, \hat{Y}_{dst}, \hat{Z}_{dst}$ i.e. we define:

$$\hat{X}_{dst} \equiv X_{dst} \,|\, \psi, \hat{Y}_{dst} \equiv Y_{dst} \,|\, \psi, \hat{Z}_{dst} \equiv Z_{dst} \,|\, \psi \quad ....(8)$$

To completely specify morphing, we still need to fill the values of the destination geometric maps for all points in $\overline{\psi} = dom_k \setminus \psi$. In other words, we need to specify the destination 3D vertices for all points of L_k in $\overline{\psi}$. Since these points do not have a physically corresponding point in L_{k+1}, we cannot apply (3) to get a destination 3D vertex from model . L_{k+1}. The simplest solution would be that no geometric morphing is applied to these points and that their destination vertices just coincide with their L_k vertices. However, in that case:

* points in ψ will have destination vertices from L_{k+1},

* while points in $\overline{\psi}$ will have destination vertices from L_k,

The problem resulting out of this situation is that the produced destination maps $X_{dst}, Y_{dst}, Z_{dst}$ (see Figs. 10 , 11) will contain discontinuities along the boundary (say $\partial\overline{\psi}$) between regions ψ and $\overline{\psi}$, causing this way annoying discontinuity artifacts (holes) in the geometry of the "morphable 3D-model" during the morphing procedure. This will happen because the geometry of both L_k and L_{k+1}, as well as their relative pose, have been estimated only approximately, and therefore these two models may not match perfectly when placed in a common 3D coordinate system.

The right way to fill in the destination vertices at the points in $\overline{\psi}$ is based on the observation that a physically valid destination 3D model should satisfy the following 2 conditions:

1. On the boundary of $\overline{\psi}$, no discontinuity in 3D structure should exist, i.e. the unknown values of $X_{dst}, Y_{dst}, Z_{dst}$ along the boundary $\partial\overline{\psi}$ should match the corresponding known values specified by $\hat{X}_{dst}, \hat{Y}_{dst}, \hat{Z}_{dst}$ along that boundary.

2. In the interior of $\overline{\psi}$, the relative 3D structure of the initial L_k model should be preserved.

Intuitively, these two conditions simply imply that, as a result of morphing, vertices of L_k inside $\overline{\psi}$ must be deformed without distorting their relative 3D structure so as to seamlessly match the 3D vertices of L_{k+1} along the boundary of $\overline{\psi}$. In mathematical terms the first condition obviously translates to:

$$X_{dst}|_{\partial\overline{\psi}} = \hat{X}_{dst}|_{\partial\overline{\psi}}, Y_{dst}|_{\partial\overline{\psi}} = \hat{Y}_{dst}|_{\partial\overline{\psi}}, Z_{dst}|_{\partial\overline{\psi}} = \hat{Z}_{dst}|_{\partial\overline{\psi}}$$

….(9)

while the second condition, which imposes the restriction of preserving the relative 3D structure of L_k, simply implies

$$\begin{bmatrix} X_{dst}(p)-X_{dst}(p') \\ Y_{dst}(p)-Y_{dst}(p') \\ Z_{dst}(p)-Z_{dst}(p') \end{bmatrix} = \begin{bmatrix} X_k(p)-X_k(p') \\ Y_k(p)-Y_k(p') \\ Z_k(p)-Z_k(p') \end{bmatrix}, \forall p, p' \in \overline{\psi}$$

…………..(10)

this is easily seen to be equivalent to:

$$\begin{bmatrix} \nabla X_{dst}(p) \\ \nabla Y_{dst}(p) \\ \nabla Z_{dst}(p) \end{bmatrix} = \begin{bmatrix} \nabla X_k(p) \\ \nabla Y_k(p) \\ \nabla Z_k(p) \end{bmatrix}, \forall p \in \overline{\psi} \qquad ….(11)$$

We may then extract the destination vertices by solving 3 independent minimization problems (one for each of $X_{dst}, Y_{dst}, Z_{dst}$) which is all of the same type. It therefore suffices to consider only one of them. E.g. for estimating Z_{dst} we need to find the solution to the following optimization problem:

$$\min_{Z_{dst}} \iint_{\overline{\psi}} \|\nabla Z_{dst} - \nabla Z_k\|^2, \text{ with } \quad Z_{dst}|\partial\overline{\psi} = \hat{Z}_{dst}|\partial\overline{\psi} \ (12)$$

For discretizing the above problem we can make use of the underlying discrete pixel grid. To this end, we assume a 4-system neighborhood for the image pixels and we denote by N(p) the corresponding neighborhood of pixel p. In this case, the boundary $\partial\overline{\psi}$ equals the set $\partial\overline{\psi} = \{p \in \overline{\psi} : N(p) \cap \overline{\psi} \neq 0\}$and the finite difference discretization of (5) yields the following quadratic optimization problem:

$$\min_{Z_{dst}} \sum_{p\in\overline{\psi}} \sum_{q\in N(p)} (Z_{dst}(p) - Z_{dst}(q) - [Z_k(p) - Z_k(q)])^2 \text{ with } \quad Z_{dst}(p) = \hat{Z}_{dst}(p), \quad \forall p \in \partial\overline{\psi} \quad \text{---} \ldots\ldots\ldots\ldots\ldots\ldots (13)$$

This quadratic problem is, in turn, equivalent to the following system of linear equations:

$$|N(p)| Z_{dst}(p) - \sum_{q\in N(p)} Z_{dst}(q) = \sum_{q\in N(p)} (Z_k(p) - Z_k(q)) \quad \forall p \in \overline{\psi}$$

$$Z_{dst}(p) = \hat{Z}_{dst}(p), \ \forall p \in \partial\overline{\psi} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (14)$$

than can be solved with an iterative algorithm very efficiently due to the fact that all these linear equations form a sparse (banded) system.

Also, an alternative way of solving our optimization problem in (5) is by observing that any function minimizing (5) is also a solution to the following Poisson equation with Dirichlet boundary conditions [11]:

$$\Delta Z_{dst} = div(\nabla Z_k) \text{ with } Z_{dst}|_{\partial\overline{\psi}} = \hat{Z}_{dst}|_{\partial\overline{\psi}} \ \ldots\ldots (15)$$

Therefore, in this case, in order to extract the geometric maps $X_{dst}, Y_{dst}, Z_{dst}$ it suffices that we solve 3 independent Poisson equations of the above type. See Figures 10 , 11 for a result produced with this method.

Conclusion

We have presented a new approach for obtaining photorealistic and interactive walkthroughs of large, outdoor scenes. To this end a new hybrid data structure has been presented, which is called “morphable 3D-mosaics". No global model of the scene needs to be constructed and at any time during the rendering process, only one “morphable 3D-mosaic" is displayed. This enhances the scalability of the method to large environments. In addition, the proposed method uses a rendering path, which is highly optimized in modern 3D graphics hardware and thus can produce photorealistic renderings at interactive frame rates. In the future we intend to extend our rendering pipeline so that it can also take into account data from sparse stereoscopic views that have been captured at locations throughout the scene and not just along a predefined path. This could further enhance the quality of the rendered scene and would also permit a more extensive exploration of the virtual environment. Moreover, this extension still fits perfectly to the current architecture of the 3D-accelerated rendering pipeline (a blending of multiple local models will still be taking place).

REFERENCES

[1]. Stan Birch_eld and Carlo Tomasi, ``Depth discontinuities by pixel-to-pixel stereo'', In the proceedings of the ICCV, 1073.1080, 1998.
[2]. M. J. Black and P. Anandan, ``The robust estimation of multiple motions: Parametric and piecewise-smooth Flow fields'', CVIU, vol. 63, No. 1, 75.104, 1996.
[3]. Jean Yves Bouguet, ``Camera Calibration Toolbox for MATLAB'', http://www .vision.caltech.edu/bouguetj/calib_doc.
[4]. Yuri Boykov and Marie Pierre Jolly, ``Interactive organ segmentation using graph cuts'', In the proceedings of the MICCAI '00: Proceedings of the Third International Conference on Medical Image Computing and Computer- Assisted Intervention, Springer-Verlag, London, UK, 276.286, 2000.
[5]. R.I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge, 2000.
[6]. David J. Heeger and James R. Bergen, ``Pyramid-based texture analysis/ synthesis'', In the proceedings of the SIGGRAPH, 229.238, 1995.
[7]. F. Heitz and P. Bouthemy, ``Multimodal estimation of discontinuous optical flow using markov random fields'', IEEE Trans. Pattern Anal. Mach. Intell., vol. 15, No. 12, 1217.1232, 1993.
[8]. David Lee and Theo Pavlidis, ``One dimensional regularization with discontinuities'', IEEE Trans. Pattern Anal. Mach. Intell., vol. 10, No. 6, 822.829, 1988.
[9]. M. Levoy and P. Hanrahan, ``Light _eld rendering'', In the proceedings of the SIGGRAPH 96, 31.42, 1996.
[10]. S. Li, Markov Random Field Modeling in Computer Vision, Springer Verlag, 1995.
[11]. X. Li and M. T. Orchard, ``Spatially adaptive image denoising under over complete expansion'', In the proceedings of the Proc. IEEE ICIP, 2000.
[12]. D Zwilliger, Handbook of Differential Equations, Boston, MA: Academic Press, 1997.

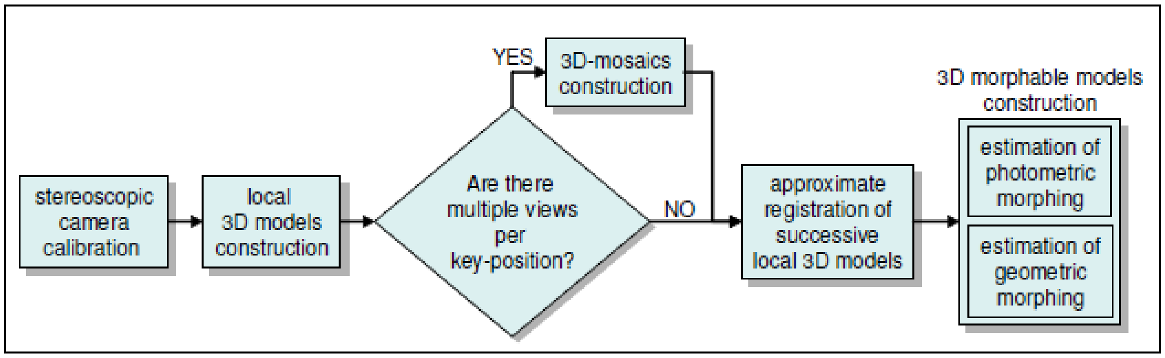

stereoscopic camera calibration
local 3D models construction
Are there multiple views per key-position?
YES
3D-mosaics construction
NO
approximate registration of successive local 3D models
3D morphable models construction
estimation of photometric morphing
estimation of geometric morphing

Figure 1: The modeling pipeline

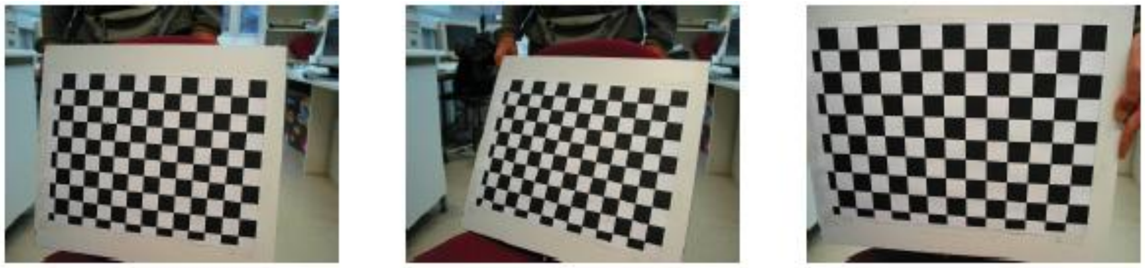


Figure 2: For calibrating our camera we capture images of a chess pattern at random positions and orientations

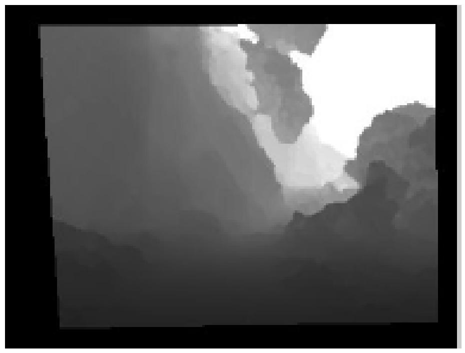


Figure 3: Depth map Z0 of a local model (black pixels do not belong to its valid region domo

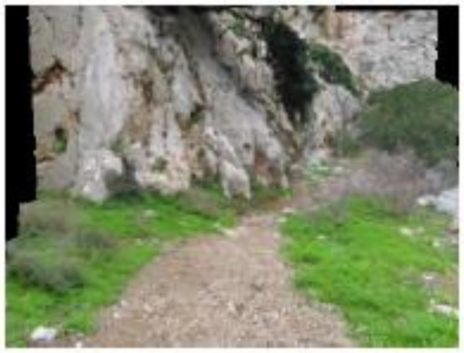


Figure 4: A rendered view of the local model using an underlying triangle mesh

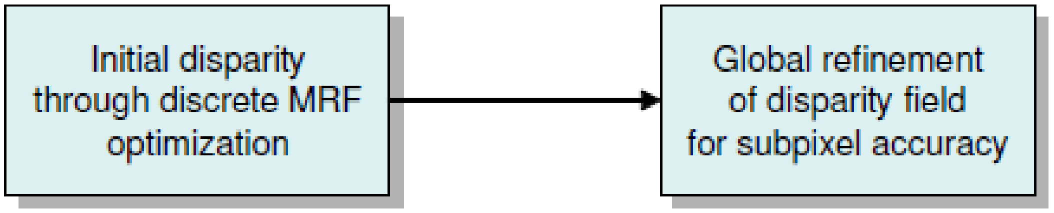

Initial disparity through discrete MRF optimization
Global refinement of disparity field for subpixel accuracy

Figure 5: The 2 stages needed for disparity estimation

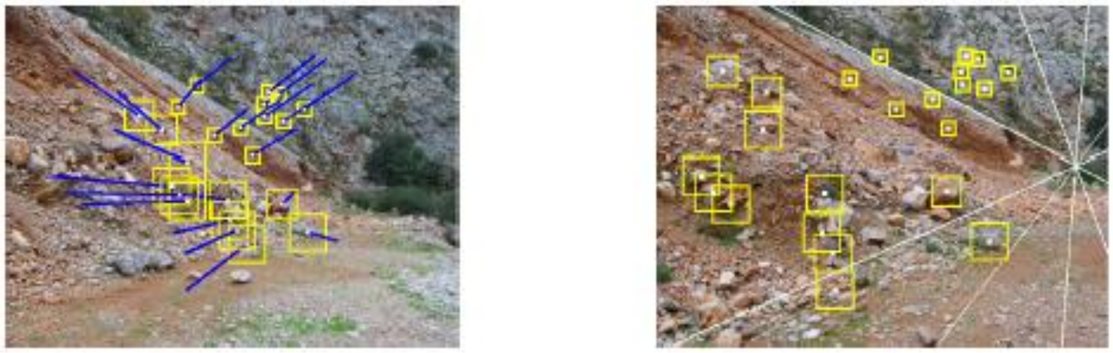


Figure 6: Images (I_K, I_{K+1})

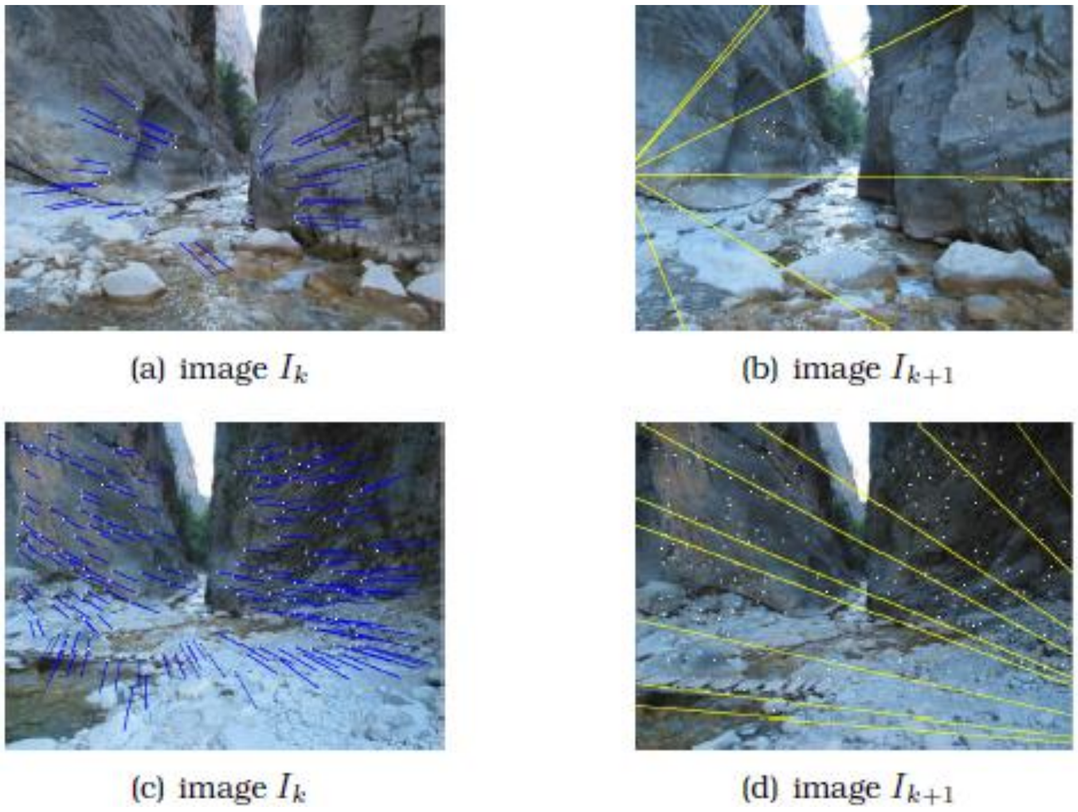


Figure 7: Optical flow vectors on images (I_K, I_{K+1})

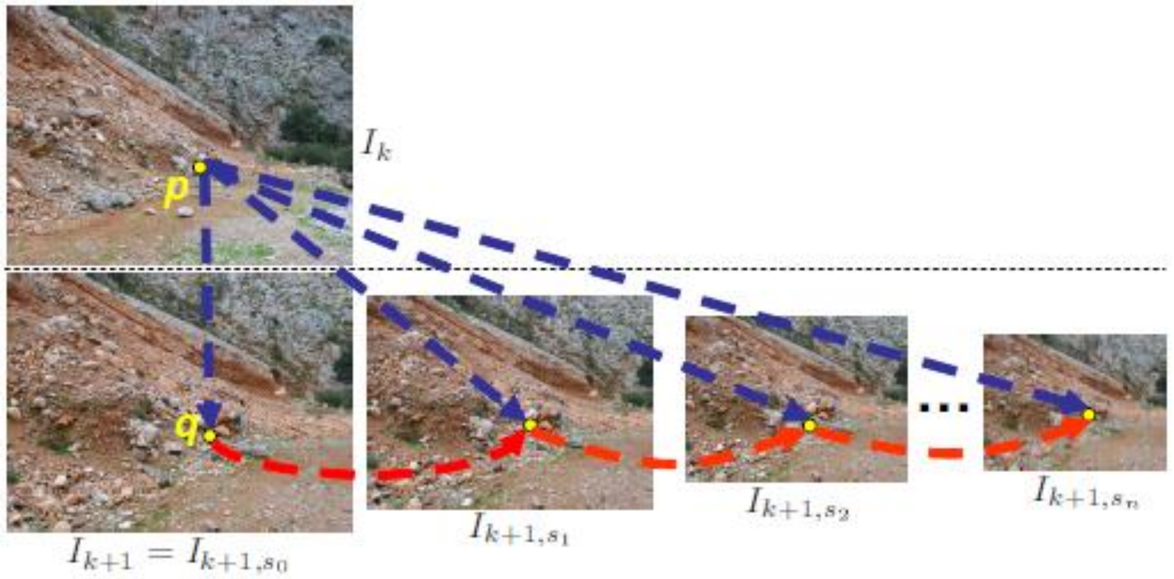


Figure 8: Resulting pixels with the neighborhood of p

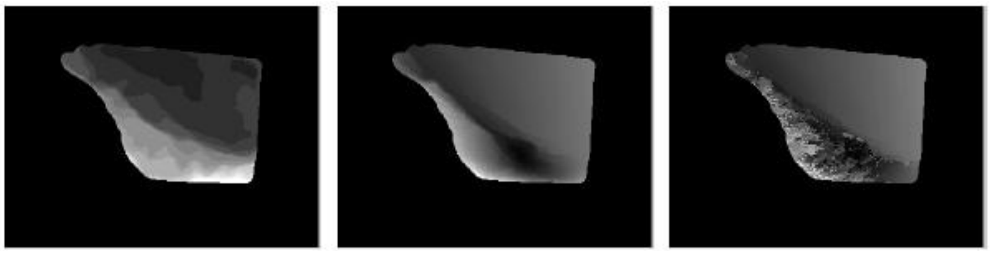


Figure 9: Scale factors and optical flow magnitudes of images

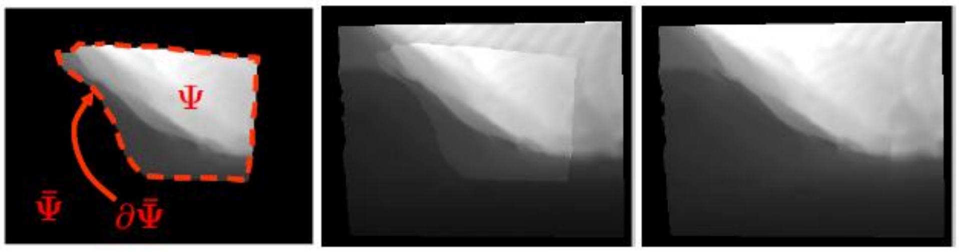


Figure 10: Destination depth map Zdst

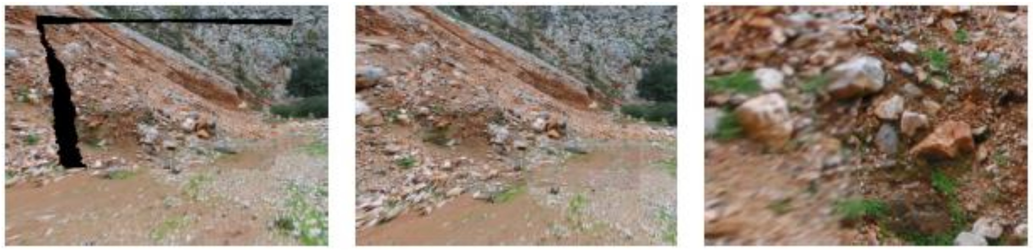
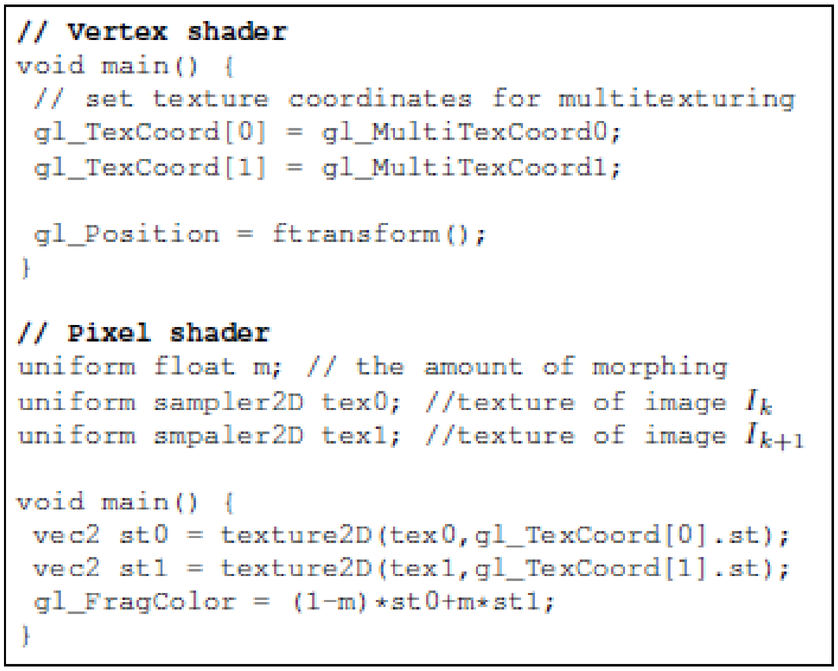


Figure 11: Rendered views of morphable 3D-model during transition from key position corresponding to image



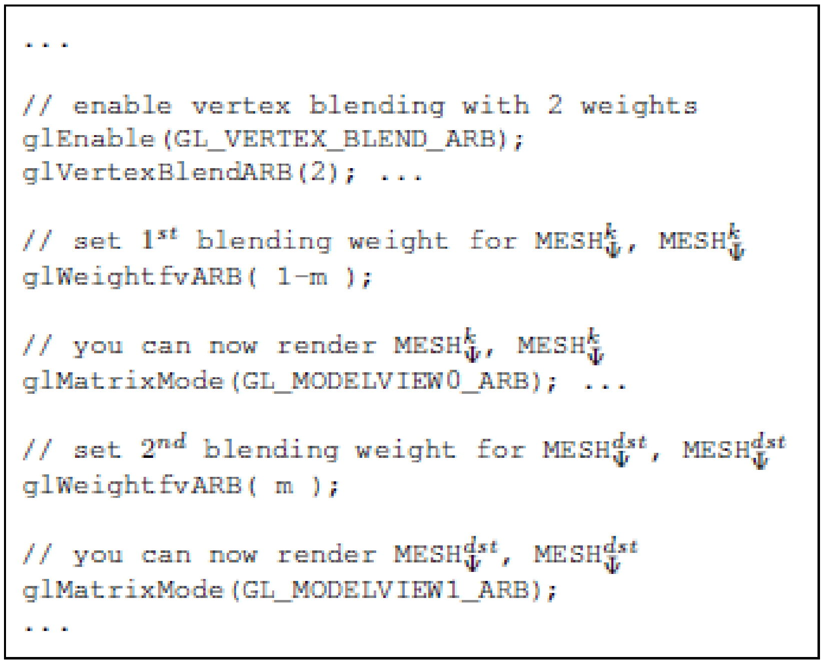
```
// Vertex shader
void main() {
 // set texture coordinates for multitexturing
 gl_TexCoord[0] = gl_MultiTexCoord0;
 gl_TexCoord[1] = gl_MultiTexCoord1;

 gl_Position = ftransform();
}

// Pixel shader
uniform float m; // the amount of morphing
uniform sampler2D tex0; //texture of image $I_k$
uniform smpaler2D tex1; //texture of image $I_{k+1}$

void main() {
 vec2 st0 = texture2D(tex0,gl_TexCoord[0].st);
 vec2 st1 = texture2D(tex1,gl_TexCoord[1].st);
 gl_FragColor = (1-m)*st0+m*st1;
}
```

Figure 12: Pixel shader code (and the associated vertex shader code), written in GLSL (OpenGL Shading Language), for implementing the photometric morphing



```
...

// enable vertex blending with 2 weights
glEnable(GL_VERTEX_BLEND_ARB);
glVertexBlendARB(2); ...

// set $1^{st}$ blending weight for MESH$^k_\Downarrow$, MESH$^k_\Downarrow$
glWeightfvARB( 1-m );

// you can now render MESH$^k_\Downarrow$, MESH$^k_\Downarrow$
glMatrixMode(GL_MODELVIEW0_ARB); ...

// set $2^{nd}$ blending weight for MESH$^{dst}_\Downarrow$, MESH$^{dst}_\Downarrow$
glWeightfvARB( m );

// you can now render MESH$^{dst}_\Downarrow$, MESH$^{dst}_\Downarrow$
glMatrixMode(GL_MODELVIEW1_ARB);
...
```

Figure 13: Skeleton code in C for applying vertex blending in OpenGL

International Association of Research Scholars

IARS

Certificate of Recognition

This certificate is awarded to

Rabi Narayan Satpathy

in recognition of his contribution

"Formation of Morphable 3D-model of Large Scale Natural Sites by Using Image Based Modeling and Rendering Techniques"

to Vol. 01, No. 02, 2011 *of*

IARS International Research Journal

[illegible]

Editor in Chief

IARS International Research Journal
29 AUG 2011